REMARKS

Examiner's Response to Previous Arguments

The Examiner's withdrawal of the rejection of claims 1 and 31 under 35 USC § 102(e) based on U.S. Patent No. 6,216,224 to Klein ("Klein") is appreciated. Because none of the pending claims are rejected under 35 USC § 102 in the latest Office Action, it appears that the withdrawal of the rejection under 35 USC § 102 applies not just to claims 1 and 31 but to all of the pending claims. This is appreciated and will be assumed to be correct unless the Examiner indicates otherwise.

Claim Rejections 35 U.S.C. § 103

Claims 1, 2, 5-9, 14, 16-19, 22-23, 26-27, 29 and 31-32 are rejected under 35 U.S.C.103(a) as being unpatentable over Klein and U.S. Patent No. 5,410,699 to Bealkowski ("Bealkowski")

The independent claims rejected as being obvious by Klein in view of Bealkowski are claims 1, 14, 22, and 31. The remaining claims depend from one of these base claims.

The Examiner appreciates that Klein alone does not anticipate any of these claims. It is kindly asserted that the combination of Klein and Bealkowski also does not render any of these claims obvious. The Examiner uses Bealkowski to fill in the missing limitations of Klein.

Claim 1 recites, in pertinent part, "an application and file storage device configured to read and write data files, one or more of the data files including the basic input/output system (BIOS) interface;...and a loading logic circuit that copies a portion of the BIOS from the storage device into the RAM".

As understood by the Examiner, Klein does not teach either of these two limitations. Klein teaches copying or "shadowing" from ROM (a type of code storage device) to RAM, not from an application and file storage device, as required by the claim. As will be discussed below, Bealkowski does not remedy the shortcomings of Klein and cannot properly be used in combination with Klein to fill in the missing limitations.

Claim 14 recites, in pertinent part, "resetting operation of a microprocessor; and thereafter suspending operation of the microprocessor; and thereafter copying a portion of a BIOS from an application and file storage device into RAM; and thereafter starting operation of the microprocessor."

As understood by the Examiner, Klein does not teach these limitations. To the contrary, Klein teaches copying or "shadowing" from ROM (a type of code storage device) to RAM, not from an application and file storage device, as required by the claim. As will be discussed below, Bealkowski does not remedy the shortcomings of Klein and cannot properly be used in combination with Klein to fill in the missing limitations.

Claim 22 recites, in pertinent part, "storing the interface in an application and file storage device; and thereafter copying the interface from the application and application and file storage device into RAM without using a microprocessor."

As understood by the Examiner, Klein does not teach these limitations. To the contrary, Klein teaches storing in and copying from a ROM (a type of code storage device) to RAM, not from an application and file storage device, as required by the claim. As will be discussed below, Bealkowski does not remedy the shortcomings of Klein and cannot properly be used in combination with Klein to fill in the missing limitations.

Claim 31 recites, in pertinent part, "an application and file storage device having a plurality of files...; and an interface for communicating between the microprocessor, the application and file storage device and the human interface devices, the interface residing in a file of the file storage device; and means for copying a portion of the interface into the random access memory without using the microprocessor." As will be discussed below, Bealkowski

does not remedy the shortcomings of Klein and cannot properly be used in combination with Klein to fill in the missing limitations.

"When a rejection depends on a combination of prior art references, there must be some teaching, suggestion, or motivation to combine the references." In re Rouffet, 149 F.3d 1350, 1355 (Fed. Cir. 1998). Stated another way, the prior art as a whole must "suggest the desirability" of the combination. In re Beattie, 974 F.2d 1309, 1311 (Fed. Cir. 1992) (internal quotation omitted); Winner Int'l Royalty Corp. v. Wang, 202 F.3d 1340 (Fed. Cir. 2000). "A reference may be said to teach away when a person of ordinary skill, upon [examining] the reference, would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken by the applicant." Para-Ordnance Mfg. v. SGS Importers Int'l, 73 F.3d 1085, 1090, 37 USPQ2d 1237, 1241 (Fed. Cir. 1995) (quoting In re Gurley, 27 F.3d 551, 553, 31 USPQ2d 1130, 1131 (Fed. Cir. 1994)).

One of skill in the art would not combine Bealkowski with Klein for a number of reasons.

First, the present application discourages following the path set out in both Klein and in Bealkowski. In particular, the present application discourages usage of a ROM for storing the BIOS, and also therefore discourages or teaches away from the practice of shadowing the BIOS from ROM to RAM, as is taught in both Klein and Bealkowski, as will be discussed below.

Second, Klein *discourages* following the teachings of Bealkowski. In particular, Bealkowski teaches using the processor to shadow the BIOS from ROM to RAM, whereas Klein specifically describes the *disadvantages* of using the processor to control these operations.

Third, the present application also teaches away from using the processor in the booting process, as taught by Bealkowski.

Each of these reasons, in its own right, indicates that the combination of Klein and Bealkowski cannot properly render the claims of the present invention obvious. Not only is there no motivation to combine the references, one of skill in the art would be actively dissuaded from combining the teachings, and would not arrive at the claimed invention.

Addressing the first point, Bealkowski, like the prior art described in the present patent application, teaches shadowing the bios from ROM to RAM before execution. In the present application, a ROM is taught to be a type of Code Storage Device (CSD). Page 2, line 12. The requirement for a ROM or other type of CSD for storing the boot code is eliminated in the present application, unlike in Bealkowski.[1] Therefore, the present invention teaches away from the solution in Bealkowski. Instead, the present application uses an Application and File Storage Devices (AFSD) that is already present in order to store the operating system and applications of a device. Code Storage Devices are distinguished from Application and File Storage Devices (AFSDs) in the present application. See pages 1 - 2. In summary, the present invention teaches using an AFSD in lieu of a ROM to store an input/output system such as the BIOS.

An AFSD is a non-volatile storage device such as solid state memory or a magnetic or optical drive. This is different from a CSD, another type of non-volatile memory, as seen on page 2 of the application:

> [r]eferring to Fig. 1, in a generic computing environment, two types of non-volatile storage devices are used. These non-volatile storage devices are used for code storage and for application and file storage. Typically a code storage device (CSD) 45 has a much smaller capacity than an application and file storage device (AFSD) 40. In addition, a CSD is usually accessed after system power up or after system reset and its content is very

[1] It should be noted that there could, of course, be a ROM or "CSD" for various other reasons, and that the *claims* do not specifically *exclude* the use of a ROM in various processes. In all comparisons, the exact language of the various different claims dictates the scope of the claims, and these portions of the specification brought to the attention of the Examiner should not be interpreted as any type of disclaimer (or other narrowing estoppel) of the scope of the claims because they are not intended as such.

> infrequently updated by the computer system in comparison to an AFSD that is updated frequently by the applications or users of the operating system.
>
> Traditionally the CSD 45 is hardwired to the system bus and is mapped into a specific memory location. After the completion of system power up reset or system reset, the CPU 30 will look for its initialization code in this specific location of CSD 45. This initialization code is dependent on system architecture or hardware, and in PC architecture is known as the BIOS. In other architectures this initialization code is referred to as initialization firmware, boot firmware etc... Examples of CSDs are EPROM, Flash ROM, and OTP PROM. An AFSD is used in order to store an operating system, application programs or general file and user data. An AFSD is a non volatile storage device such as solid state memory or a magnetic or optical drive
>
> The BIOS or booting firmware is normally stored in the CSD 45 in order for the CPU to execute its instructions. Traditionally if boot code is stored in the AFSD 40, a small set of instructions, constituting the basic BIOS must be stored in the CSD 45 in order to copy the BIOS or booting firmware into system RAM 50. In this invention, the requirement of the CSD for storing and executing these initialization instructions is eliminated. Page 2, lines 2-20.

This description of the prior art from the Background section of the present application, which the present application teaches away from, is similar to the system taught by Bealkowski. While the present invention eliminates the requirement of the ROM (a type of CSD) for storing the BIOS, Bealkowski teaches usage of a ROM for this very purpose. Bealkowski teaches using *both* a ROM and a fixed disk or diskette to store the BIOS. Bealkowski teaches storing a first portion of the BIOS in ROM and a second portion of the BIOS in a hard disk or a diskette. In the present application, Bealkowski's first portion is referred to as a type of basic BIOS, and the second portion that is copied from the fixed disk or diskette is referred to as the BIOS and/or the BIOS Image. This is seen in the present application's discussion of the prior art, a portion of which is excerpted above.

In Bealkowski, the (first) portion of the BIOS in the ROM is executed before the rest of the BIOS can be fetched from the fixed disk or diskette. The instructions contained in the (first) portion of the BIOS in the ROM cause the remaining portion of the BIOS stored in the fixed disk or diskette to be loaded. This is known in the art as "bootstrapping."

> Previous to the present invention, ROM 36 could include all of the BIOS code which interfaced the operating system to the hardware peripherals. According to one aspect of the present invention, however, ROM 36 is adapted to store only a portion of BIOS. This portion, when executed by the system processor 26, inputs from either the fixed disk 62 or diskette 66 a second or remaining portion of BIOS, hereinafter also referred to as a BIOS image. Bealkowski Col. 6, lines 53-61.

In addition to teaching bootstrapping, Bealkowski is focused upon creating a system that can load the second or "bootstrapped" portion of the BIOS from a diskette in the case that the hard disk becomes incapacitated. Id.; See also Fig. 4. This requires that the first portion of the BIOS that is stored on the ROM be executed before the remainder of the BIOS can be loaded from the diskette. This is the nature of a "bootstrapping" type system, as taught in Bealkowski. Bealkowski teaches a priority mode and a recovery mode, both of which involve "bootstrapping" from ROM.

> The read only memory includes a first portion of BIOS which in operation, initializes the system and detects the state of the switching means electrically coupled to the system processor. If the switching means is in a priority position, a master boot record is immediately loaded from the diskette drive (priority mode) without testing for the fixed disk drive. If the switching means is in a recovery position the first portion of BIOS attempts to load the master boot record from the fixed disk. If the fixed disk is not available, non-operational, or the master boot record on the disk is invalid, the first portion of BIOS reads in the master boot record from the diskette drive (recovery mode). Col. 3, lines 27-40.

In summary regarding the first point, Bealkowski and Klein teach usage of a ROM to store the BIOS, whereas the present application *teaches away from* usage of the ROM (a type of CSD) to the BIOS (PC boot code). Thus, for this reason alone, the combination of Klein and Bealkowski cannot properly form the basis of an obviousness rejection.

Turning now to the second point, Klein and Bealkowski cannot properly be combined to render the claims (which each have unique limitations that were recited above) obvious because Klein teaches away from Bealkowski. Bealkowski teaches that the initial BIOS load ("IBL") routine is controlled by the system processor. The IBL includes the mater boot record ("MBR") code that is executed by the system processor. See Bealkowski Col. 7, line 45 to Col. 8 line 12; see also FIG. 4. Klein discusses at length why using a processor in the booting or bootstrapping process, as is taught by Bealkowski, is a bad idea. The relevant portion of Klein is reproduced below.

> Although ROM shadowing vastly increases the speed of PC initialization by speeding up subsequent execution of firmware routines, transfer of the firmware routines to the system RAM 116 in Intel Pentium CPU-based PCs is inefficient. The inefficiencies in ROM shadowing arise largely because the Intel Pentium CPU is not yet initialized when it carries out ROM shadowing. As one result, the instruction cache (not shown) is not available within the CPU 106, requiring the CPU 106 to repeatedly fetch and execute the instructions of a small loop of instructions that transfer the firmware routines from the ROM 104 to the RAM 118. Each instruction is loaded from the ROM 104, via at least one relatively low-speed bus, into a register and then executed in a second step. When a CPU 106 has been initialized, and instruction caching is available, the instructions of a small loop of instructions are fetched only once from the RAM during the first iteration of the loop. The instructions are stored in the CPU instruction cache as they are fetched. When the instructions are needed in a subsequent iteration of the loop, they are fetched from the instruction cache, rather than from the RAM. Instructions in the CPU instruction cache can be fetched more quickly than instructions in memory because, by fetching from the instruction cache, access of

instruction through the system controller 114 and either the memory bus 120, in the case of instructions fetched from the RAM 118, or the ISA 108 and PC 112 buses, in the case of instructions fetched from the ROM 104, is avoided. However, in an uninitialized Pentium CPU, caching is not available, and therefore each instruction of the loop of instructions that transfer the firmware routines from the ROM 104 to the RAM 118 must be fetched from the ROM 104 prior to execution during each iteration of the loop.

Intel Pentium processors prefetch instructions. Instruction prefetch allows a CPU to simultaneously execute one instruction while fetching the next instruction from memory. Instruction prefetch is very efficient for executing a linear, sequential series of instructions. It is also efficient when the CPU can correctly predict which of two possible instructions following a branch instruction will be next executed. However, if the instruction being executed causes the processor to branch to an instruction other than the predicted instruction, the instruction being incorrectly prefetched from memory will then need to be discarded from an internal cache within the CPU. CPU execution is stalled until the instruction that is the target of the branch instruction is read from memory by the CPU.

An uninitialized CPU cannot perform effective branch prediction. A second result of carrying out ROM shadowing with an uninitialized CPU is that prefetching will, in a large percentage of cases, obtain the wrong instruction. The loop of instructions used to transfer the firmware routines from the ROM 104 to the RAM 118 is a very tight loop in which a unit of data is transferred from the ROM to the RAM, a counter is decremented, and a branch instruction is executed to return control back to the top of the loop. Thus, a large proportion of the instructions executed during ROM shadowing are branch instructions. Because effective branch prediction cannot be carried out by an uninitialized CPU, the tight loop of instructions used in ROM shadowing effectively defeats the prefetch strategy of the CPU and further slows the ROM shadowing process. A need has therefore been recognized for a way to transfer firmware routines from the ROM to the RAM during PC initialization that does not require execution of a loop of instructions by the CPU. Klein Col. 2 line 19 to Col. 3 line 16 (emphasis added).

Therefore, one skilled of the art, would, contrary to the Examiner's assertions, not be motivated to combine the references by any teachings within the references, but would rather be discouraged from combining the contradictory teachings of the references.

Turning now to the third point, the teachings of the present application also teach away from using a processor to control the booting process of copying the BIOS to RAM.[2] The present invention, rather, teaches holding or halting the operation of the processor, and instead using the claimed logic loading circuitry rather than the processor. See page 6, lines 10-18 and FIG. 4 of the present application. FIG. 4 is reproduced on the right for the convenience of the Examiner.

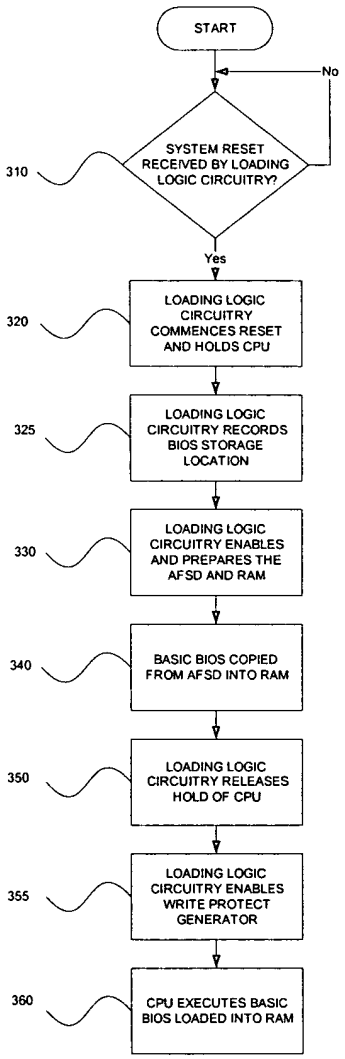


Fig. 4

The various independent claims recite different aspects and combinations of these processes and related structures, although the claims speak for themselves.[3] Therefore, the present application teaches away from Bealkowski, and one of skill in the art would not combine Bealkowski with Klein.

The dependent claims for each of the independent base claims are allowable for all the reasons above regarding the base claims. Furthermore, many of the dependent claims also contain

[2] The processor, is, of course, taught to execute the BIOS once copied into RAM, whether or not the BIOS is bootstrapped or loaded in step, while the processor is on hold.
[3] Any discussion of the teachings of the present application refer only to the teachings of the specification, and not the claims, unless specifically indicated to the contrary.

elements not disclosed by the cited art, and the specific limitations of the dependent claims are not obvious as discussed below. This is done for purposes of brevity and the arguments regarding these claims incorporated by reference in this section.

Claims 3-4, 20-21, 25 and 33 are rejected under 35 U.S.C.103(a) as being unpatentable over Klein, Bealkowski and U.S. Patent No. 5,579,522 to Christeson et al. ("Christeson")

The Examiner relies upon Christeson for the disclosures of a flash memory device and magnetic or optical drive not taught by Klein, and the combination of Klein and Bealkowski, in support of the rejection of several dependent claims listed above. As discussed in the previous response with regard to the § 102 rejection, each of the above listed dependent claims is novel and not anticipated by Klein. As discussed above in this response, these claims are also not obvious under §103 because Bealkowski cannot properly be combined with Klein to remedy the lack of teachings of Klein.

Claim 3 adds to independent claim 1 that "the application and file storage device is a flash memory device." Claim 4 adds to claim 1 that "the application and file storage device is a magnetic or optical disk drive."

Like Klein, Christeson also does not teach the claim 1 recitation of "an application and file storage device configured to read and write data files, one or more of the data files including the basic input/output system (BIOS) interface;...and a loading logic circuit that copies a portion of the BIOS from the storage device into the RAM" (emphasis added). Nor does it teach the claim 14 recitation of "copying a portion of a BIOS from an application and file storage device into RAM," and the claim 22 recitation of "storing the interface in an application and file storage device; and thereafter copying the interface from the application and file storage device into

RAM without using a microprocessor," and the claim 31 recitation of "an application and file storage device having a plurality of files...an interface for communicating between the microprocessor, the application and file storage device and the human interface devices, the interface residing in a file of the file storage device; and means for copying a portion of the interface into the random access memory without using the microprocessor."

The application and file storage device in Christeson is the data storage device 106 seen in Christeson's Figure 1. The hardware/software interface (BIOS in PC terminology) is not stored in this storage device 106 but in flash memory 103. This flash memory is a type of code storage device as distinguished on page 2, lines 2-20 of the present application, which was excerpted above in this Response. The teaching in Christeson that the code storage device can be flash memory (so that it can be easily updated without removing any parts or covers from the computer) does mean that Christeson teaches the various claim limitations above regarding an application and file storage device.

In addition to the fact that the combination of Klein and Christeson does not teach the claimed invention, there is no motivation to combine the two references. Christeson teaches away from using a loading logic circuit and therefore cannot be properly combined with Klein for all the reasons discussed above regarding Bealkowski.

Claims 20-21, 25 and 33 are similar to claims 3 and 4 and allowable for all the same reasons as claims 3 and 4.

Claim 24 is rejected under 35 U.S.C. 103(a)) as being unpatentable over Klein ("Klein"), Bealkowski, Christeson, and further in view of U.S. Published Application No. 2002/0138702 to Gefen et al. ("Gefen")

Claim 24 depends from claim 22 and adds the additional recitation "wherein the application and file storage device is a NAND flash memory device."

Claim 24 is allowable for all the reasons above regarding Klein, Bealkowski, and Christeson. For all the reasons given above, Bealkowski cannot be properly combined with Klein. Also, as discussed above with regard to claims 3-4, 20-21, 25 and 33, the combination of Klein and Christeson does not teach the various claim recitations regarding use of an application and file storage device. The addition of Gefen does not remedy the shortcomings of the previous combinations, and there is no motivation to combine Gefen with the other references.

While Geffen does teach NAND, it does not teach the booting of a system or using NAND for such purposes and one of ordinary skill in the art would not be motivated to combine Geffen with either Klein or Christeson. It is kindly asserted that the Examiner's cited motivation to combine is insufficient, as best as it can be understood. The Examiner indicates that "since the technology for implementing NAND flash memory was well known and since NAND flash memory benefits from a lower cost, is non executable, and requires less routing resources, an artisan would have been motivated to implement a NAND flash in the system of Klein and Bealkowski and Christeson." It is unclear why the fact that NAND is non-executable would motivate one to combine it with Klein, Bealkowski, and Christeson. Furthermore, it is also not clear what the Examiner means when he indicates that NAND requires less routing resources and if that is the case regarding implementation in the present invention. In general, the Examiner appears to simply take the disclosure of NAND to fill in the missing limitation, using the claim

itself as a roadmap. It is kindly asserted that this is hindsight and that it is impermissible to pick and choose missing elements from the prior art to arrive at the claimed invention.

Claims 10-11 and 34-35 are rejected under 35 U.S.C. §103(a) as being unpatentable over Klein, Bealkowski, and U.S. Patent No. 5,809,559 to Kim ("Kim")

Claim 10 depends from claim 1 and is allowable for all the reasons regarding claim 1 discussed above and in the previous response. Claim 10 adds the additional recitation "wherein the loading logic circuitry comprises a write protect mechanism that prevents the location of the storage device having the BIOS from being overwritten."

Here Kim is introduced for the proposition that a write protect mechanism was well known. However, the Examiner has not cited any motivation to combine other than the general proposition that a write protect mechanism "benefits" the BIOS by preventing corruption. Kim teaches and is directed towards a system and method utilizing a virtual addressing buffer circuit to emulate a device which is physically not present. This is a very different endeavor than the booting operation taught by the present application, and there is no logical connection that would motivate one of skill in the art to combine these references. Again, the Examiner appears to have chosen from the vast library of prior art in order to arrive at the claimed invention using impermissible hindsight.

Claim 11 is dependent from claim 10 and adds the recitation "wherein the write protect mechanism generates a first and second write strobe signal for each write strobe signal of a microprocessor."

Kim simply does not appear to teach this (at the cited location or elsewhere) and the Examiner's assertion that this is obvious is unsupported.

Claims 34-35 have similar limitations and are not obvious for the same reasons as claim 10.

Claim 15 is rejected under 35 U.S.C. §103 (a) as being unpatentable over Klein, Bealkowski, and U.S. Patent No. 6,718,464 to Cromer et al. ("Cromer")

Claim 15 is dependent from claim 14 and is allowable for all the reasons regarding the base claims from which it depends. Claim 15 adds the recitation "wherein the storage device includes multiple BIOSs and wherein a user can select which BIOS to copy from the application and file storage device into RAM."

Cromer is cited for the proposition that selecting from multiple BIOSs is well known and that one of ordinary skill in the art would therefore have been motivated to implement such a selection in the combination of Klein and Bealkowski because it would benefit by such a configuration. As discussed at length above, Bealkowski cannot properly be combined with Klein. The present application teaches away from the teachings of Cromer as Cromer appears to store the BIOS in a dedicated Code Storage Device 242 rather than an AFSD 222, which was discussed previously. See Cromer Fig. 2. Additionally, Cromer appears to use the processor to perform any shadowing, which both the present application and Klein teach away from. Therefore, the combination of Klein, Bealkowski, and Cromer cannot properly render claim 15 obvious.

Claims 28 and 30 are rejected under 35 U.S.C. §103 (a) as being unpatentable over Klein, Bealkowski, and U.S. Patent No. 6,154,838 to Le et al. ("Le")

Claim 28 is dependent from claim 22 and is allowable for all the reasons regarding the base claims from which it depends. Claim 28 adds the recitation of "copying additional interface device commands from the application and file storage device into RAM using the microprocessor."

Le is cited for the proposition that it teaches copying additional interface device commands from the applications and file storage device into RAM using the microprocessor. This is incorrect. It does not. As in Klein, the flash ROM that Le copies from is not an application and file storage device, but is a code storage device. Again, a code storage device is distinguished from an application and file storage device, as discussed on Page 2, lines 2-20 of the present application, excerpted supra.

Additionally, it is kindly asserted that the Examiner's cited motivation to combine is generic in nature and appears to rely on hindsight. The Examiner indicates that one of ordinary skill in the art would be motivated to combine Le with Klein and Bealkowski "to positively affect the economics of operating and maintaining the computer system." Office Action, page 16. It is kindly asserted that a proper motivation to combine must involve a specific teaching in the reference that would lead one of ordinary skill in the art to combine it with the other reference. Utilization of such generic benefits, in hindsight, is improper and would render virtually any computer related patent application obvious.

Claim 30 is dependent from claim 22 and is allowable for all the reasons regarding the base claims from which it depends. Claim 30 adds the limitation "wherein the step of copying the interface further comprises the usage of error correction code."

The Examiner states that Le discloses using error correction code in column 14, lines 24-39. However, there is no mention of error correction code at this location, and Le appears to teach the use of a checksum (Fig. 5), which is a very different alternative to error correction code.

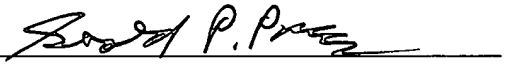
Allowable Subject Matter

The Examiner's indication that claims 12 and 13 would be allowable if rewritten in independent form is greatly appreciated. It is believed that these claims are allowable as written for the reasons discussed above.

Conclusion

Accordingly, it is believed that this application is now in condition for allowance and an early indication of its allowance is solicited. However, if the Examiner has any further matters that need to be resolved, a telephone call to the undersigned attorney at 415-318-1163 would be appreciated.

Respectfully submitted,



July 28, 2005

Gerald P. Parsons
Reg. No. 24, 486

PARSONS HSUE & DE RUNTZ LLP
655 Montgomery Street, Suite 1800
San Francisco, CA 94111
(415) 318-1160 (main)
(415) 318-1163 (direct)
(415) 693-0194 (fax)

Date